Constrained Parameter Optimization with NEURON: Simulated Annealing with Recentering

Christina M. Weaver
February 18, 2007

This document serves as a primer to the Simulated Annealing with Recentering optimization tool available for use with NEURON's Multiple Run Fitter. The general idea of the algorithm is presented, with a description of the essential parameters governing the search. The document concludes with instructions on how to install the tool.

This document assumes that the reader is familiar with NEURON's Multiple Run Fitter [4]. A tutorial on the MRF is available online [3].

1. Overview

SimAnnealRctr is a tool for constrained optimization, for use with Neuron's Multiple Run Fitter. The power of this tool is demonstrated in Ref. [10]. The algorithm is modified from the Press et al. simulated annealing algorithm for unconstrained, continuous optimization found in Numerical Recipes in C [9], and was first introduced by Cardoso et al. [2].

The simulated annealing search algorithm tests many different points in parameter space, seeking the global minimum (i.e. the point lying the furthest downhill). Uphill moves are accepted with some probability, allowing escape from local minima. This probability is proportional to a dimensionless temperature parameter, which is high at the outset of the search, and decreases as the search continues. Whenever the algorithm encounters a point in parameter space which lies outside the user-specified parameter domain, it is replaced by a point randomly selected from a neighborhood of the current minimum [2]. The size of this neighborhood is temperature-dependent.

SimAnnealRctr is adapted from Andrew Davison's SimAnneal tool [5,6], an implementation of the Press et al. algorithm for the Multiple Run Fitter.

2. Installation Instructions

The Simulated Annealing with Recentering optimization can be added to your version of NEURON without needing to recompile the code. These files have been tested for use with NEURON 5.9.39, the current standard distribution as of July 2006.

The major addition to the Multiple Run Fitter module is the 'simanneal_cardoso.hoc' file. All other files have minor changes. This distribution includes the following files:

mulfit.hoc
mulfit1.hoc

fitparm.hoc
simanneal_cardoso.hoc
eparmlst.hoc

These files must be copied into directories where your version of NEURON will know where to look for them. Before copying the files below into the specified directories, it is best to make backup copies of the original files.

The first file, mulfit.hoc, goes in $HOME/share/nrn/lib/hoc. All others belong in $HOME/share/nrn/lib/hoc/mulfit.

('$HOME' corresponds to wherever your version of NEURON sits. If this is a Windows machine, it is probably something like 'C:/nrn59'; for Unix/Linux, it may be '~/neuron/'.)

Once these files are copied into the proper directories, start up NEURON. Now, when you bring up a Multiple Run Fitter and Select an Optimizer, "Simulated Annealing with Recentering" should appear after the standard PRAXIS option.

3. Options and Parameters

The SimAnnealRctr algorithm uses the Nelder-Mead simplex algorithm [8] to determine which points in parameter space are visited during the search. The randomness of the simulated annealing approach is applied to the Nelder-Mead simplex method by adding temperature-dependent noise to the objective function values [9]. At the start of the optimization, the initial temperature is very large and nearly all tested points are accepted. As the optimization continues, the magnitude of the noise decreases towards zero. The magnitude of noise is specified by the temperature parameter, and is controlled by an annealing schedule, an equation that specified how the temperature decreases throughout the optimization.

The SimAnnealRctr algorithm uses the simple annealing schedule proposed by Kirkpatrick et al. [7], a proportional temperature decrease. At the current temperature T^j, a certain number I_t of points are tested using the simplex-simulated annealing method described above. The next temperature T^{j+1} is given by $T^{j+1} = C \cdot T^j$, where $C < 1$ is the cooling rate.

3.1 MulRunFitter options

When the user selects "Simulated Annealing with Recentering" as the Optimizer to be used by the Multiple Run Fitter, the SimAnnealRctr interface appears. The first half of the window shows various parameters of the generic MulRunFitter class. Of particular interest is the checkbox "Append the path to output file". If this box is checked, a record of minimum values found by the search algorithm will be written to a file called "OUTNAME.tmp", where OUTNAME is the string that appears below the Append checkbox. To change the name of this output file, click on the "Change output filename"

button and enter the desired file basename. For the description of the remaining MulRunFitter options, see the MulRunFitter reference page [4].

When the optimization is complete, the contents of OUTNAME.tmp are copied into OUTNAME.fit. The auxiliary file OUTNAME.info includes descriptions of the columns of the .tmp and .fit files.

3.2 SimAnnealRctr options

The second half of the SimAnnealRctr window shows parameters that are specific to the Simulated Annealing/Boundary Recentering algorithm.

The parameters alpha, beta, gamma, and lambda specify the magnitude of the reflection, expansions, and contractions of the simplex method. For a description, see Press et al. [9]; default values should provide satisfactory results for most problems.

The search result is highly sensitive to the choice of the parameters controlling the annealing temperature schedule:

- Initial temperature T_0
- Cooling rate C
- Number of iterations per cooling step I_t

If the simulated annealing method is not giving satisfactory results from several initial conditions, increases of T_0, C, and I_t may be necessary.

The cooling rate C should be a number close to (but must be less than) 1; 0.9 or 0.95 are reasonable starting values.

A large number of iterations per cooling step is important, to allow the search method to obtain a reasonable sample of the parameter space. For optimization of N parameters, a good starting point is $100 \times N$. If the method does not seem to converge, try increasing this number.

To estimate a reasonable value for the initial temperature, click the "Estimate initial temp" button on the SimAnnealRctr window. The estimation procedure carries out It iterations of the search algorithm, assuming a very large initial temperature. The number m_1 of points accepted by the search procedure is counted, as well as the number m_2 of rejected points. In addition, the average increase of the function for the m_2 rejected moves is determined. The initial temperature is then estimated by the equation

$$X = \frac{m_1 + m_2 \exp\left(\frac{-\Delta f^+}{T_0}\right)}{m_1 + m_2}$$

where X, the ratio of accepted points, is 0.95 [1,2]. The estimate for T_0 is written to OUTFILE.est.

This only provides one estimate of a suitable initial temperature. To improve the estimate, repeat this procedure from a number of initial points in parameter space, and take the maximum result as T_0. If the search algorithm does not seem to converge from this initial temperature, T_0 may have to be increased further.

Other parameters for the search algorithm include ftol, Norm_grad_tol, and min_temp. When the difference between the best and worst points of the simplex is less than ftol, or the gradient of recent visited points is less than Norm_grad_tol, or the annealing temperature is below min_temp, the search algorithm terminates.

Finally, there is also a legacy “restart” feature remaining from Andrew Davison’s SimAnneal implementation (quoted from the SimAnneal documentation [5]). This is different from the restart capability described below (Sec. 1.4). With this feature, the lowest-ever point may be reintroduced into the Simplex (called a `restart') whenever the temperature has been reduced by a factor "Restart factor". If "Restart factor" is negative, restarting is not used.

3.3 Restarting the search if your computer quits

An optimization may take days to run if the problem is difficult enough. Sometimes your computer may halt before the optimization is finished, e.g. if power to the machine is lost. In such cases the user can restart the optimization near where it left off.

Suppose you are running an optimization, and have told NEURON that you want to save the output to files with the basename SAVEPATH. In this case, at the beginning of every temperature step, a file named SAVEPATH.rst is written. This file contains all vertices of the current simplex and the current temperature.

At the start of every temperature step, all points of the simplex are written to the file OUTFILE.rst. To restart an optimization from the beginning of the last recorded temperature step, the user should check the box marked “Restart from .rst file.” (This is different from the “restart” mechanism retained from the Davison code [5]).

Bring up the Simulated Annealing Menu. On the bottom half of the menu, click on the button marked “Restart from .rst file”. Click on the button marked Change “restart filename” and enter the filename "SAVEPATH.rst", where SAVEPATH is whatever output file basename you used when you ran the halted optimization.

To restart the optimization, click on the “Optimize” button on the top half of the menu. The SimAnnealRctr tool will now set the initial values of the temperature and the simplex points from the data stored in the specified .rst file. All other optimization parameters must be the same as in the original optimization. The restart mechanism does NOT check that the current parameters match the previous one, so make sure they are correct.

The new optimization will continue at the start of the temperature step where the original job was terminated.

3.4 Search in log space

As with the PRAXIS optimization method, the user can opt to transform all parameters into log space. This is very helpful when the parameters to be optimized vary over several orders of magnitude.

4. Output files

To write the search data to output files, be sure that the checkbox "Append the path to output file" is checked. *If it is not checked, no output files will be written.* If this box is checked, output will be written to four files: OUTNAME.tmp, OUTNAME.fit, OUTNAME.info and OUTNAME.ann, where OUTNAME is the string that appears below the Append checkbox. To change the name of this output file, click on the "Change output filename" button and enter the desired file basename. The content of each of these files is described below.

- **OUTNAME.tmp:** Written while the search progresses. Each time a new minimum is found, it is recorded here. The current minimum is also recorded right before the annealing temperature is reduced during the search.

- **OUTNAME.fit:** When the simulated annealing search is over, the data contained in OUTNAME.tmp is copied to this file. It is given a header specifying how many columns are contained in each row of the data.

- **OUTNAME.info:** This file gives information about the search, including descriptions of the columns of the .tmp and .fit files.

- **OUTNAME.ann:** It can be useful to have some idea of all the points that are visited during the search. This helps to identify whether the search is focused on a particular region of parameter space, or whether the entire space is being sampled somewhat equally. Rather than reporting ALL of the points visited during the search, a subset of these points is recorded. At the end of each annealing temperature step, the current point is written to the OUTNAME.ann file. **This is not the best point found so far**; it is only a record of what part of parameter space is being sampled at this time during the search.

These files do not get overwritten when a new search is performed with the same file basename. New runs are appended to the end of the previous file. To better distinguish among different optimization runs, it is recommended to use a different file basename for each new search (e.g. select "Change output filename" and enter 'OUTNAME2').

References

[1] Aarst, E. and Korst, J., *Simulated annealing and boltzmann machines: a stochastic approach to combinatorial optimization and neural computers.* New York: Wiley, 1989.

[2] Cardoso, M.F., Salcedo, R.L., and de Azevedo, S.F., The simplex-simulated annealing approach to continuous non-linear optimization. *Computers Chem. Engng.* **20**(9): 1065-1080, (1996).

[3] Carnevale, N.T. and Hines, M.L., Using NEURON's Optimization Tools. http://www.neuron.yale.edu/neuron/docs/optimiz/main.html, (2004).

[4] Carnevale, N.T. and Hines, M.L., MulRunFitter. http://www.neuron.yale.edu/neuron/docs/help/neuron/stdrun/mulfit.html#MulRunFitter, (2006).

[5] Davison, A.P., *Personal communication*, 2004.

[6] Davison, A.P., Zhou, Z., Hines, M.L., and Shepherd, G.M., Simulating sodium and potassium currents in an olfactory mitral cell model. *Society for Neuroscience Abstracts*. **28**: 312.311, (2001).

[7] Kirkpatrick, S.C., Gelatt, D., and Vechi, M.P., Optimization by simulated annealing. *Science*. **220**: 671-680, (1983).

[8] Nelder, J.A. and Mead, R., A simplex method for function minimization. *Comput. J.* **7**(4): 303-313, (1965).

[9] Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P., *Numerical Recipes in C: The Art of Scientific Computing*. Second ed. Cambridge, U.K.: Cambridge University Press, 1992. 1020.

[10] Weaver, C.M. and Wearne, S.L., The role of action potential shape and parameter constraints in optimization of compartment models. *Neurocomputing*. **69**: 1053-1057, (2006).